Demo: Creating Sandboxed User Environments with Jupyterhub and Docker

Shreyas Cholia*, Donald Winston, Daniel Gunter
Lawrence Berkeley National Laboratory
*1 Cyclotron Road, Berkeley, CA 94720, USA; email: scholia@lbl.gov

Abstract: *In this demo we would like to showcase an infrastructure for sandboxed notebook platforms using Docker and Jupyterhub. We will show how this infrastructure is used for two scientific applications: in The Materials Project, we created a per-user sandboxed environment for workshops and tutorials, that can be directly accessed over the web; for the Institute for the Design of Advanced Energy Systems (IDAES), we enabled playgrounds for users where they can explore scientific models and data, without the difficulties of going through a complex installation and setup process.*

1. Case Studies

1.1 Materials Project: Jupyter for Workshops

For the tutorial sessions of a recent Materials Project Workshop we used Jupyterhub to provision and manage executable notebook environments. We created docker containers with the entire materials project software stack pre-installed, along with a MongoDB service and a port-mapping scheme that allowed users to locally host and inspect project-relevant web services.

We also provided tutorial notebooks, with worked examples and exercises. This allowed users to follow along during the tutorial and interact with code examples.

The workshop had over 50 users. We were able to achieve scalability via Docker Swarm. Jupyterhub provisioned notebooks as pre-defined Docker containers; as we added new users and needed additional compute power we simply added new nodes to the Swarm. This was completely transparent to Jupyterhub and allowed us to seamlessly scale up.

Presented at Gateways 2016, San Diego Supercomputer Center, La Jolla, CA, November 2-3, 2016.
https://gateways2016.figshare.com/

1.2 IDAES: Jupyterhub for Persistent Sandboxes

The Institute for the Design of Advanced Energy Systems (IDAES) stack has a tricky installation process, which includes a combination of complex software dependency management, and custom modifications to software. Scientists on the project would like to be able to experiment with different models, visualize results and develop code in a sandboxed interface while sharing their results with others. Using Docker, we were able to capture a complex execution environment and make it available as a Jupyter notebook.

The Jupyterhub environment provides the ideal system for doing this. Users are able to login to the notebook environment with their Github credentials (via the Jupyter OAuth module) and commission notebooks with a fully installed environment. The DockerSpawner module of Jupyterhub spins up new docker containers for each user with this environment.

Each user now has a persistent Jupyter notebook sandbox directly accessible from their web browser, and can iterate on their scientific models as needed. We provide a common filesystem layer for users to share their notebooks with other users on the system. Since the Jupyter notebook interface also provides a UNIX terminal and a text editor, users are able to do all their work directly inside this environment.

2. Demo

The overall setup process for both these use cases is very similar. In our demo, we will showcase the following:

- Creating a multi-user notebook environment with Jupyterhub
- Enabling user authentication with Github and OAuth
- Using Docker containers to create pre-installed sandboxed environments

- Deploying and executing these Docker-ized notebook environments
- Accessing the environment through the Jupyter terminal
- Starting up additional services in the container
- Scaling up with Docker Swarm
- Sharing notebooks and data products on this platform

We plan to show the end-to-end process for such a setup, and how it could be generally applicable to any docker-based cluster environment. We will be making our code and containers freely available so that others can reproduce this work, more or less out of the box. There will also be some discussion about how one could use easily replace certain components like authentication (i.e. replacing Github with other OAuth providers or PAM authentication) or cluster management (i.e. replacing Swarm with Kubernetes), through simple configuration file changes.

3. References

[1] Docker, https://www.docker.com/

[2] Jupyter, http://jupyter.org/

[3] Perez, Fernando. "IPython: From interactive computing to computational narratives." *2015 AAAS Annual Meeting (12-16 February 2015)* 15 Feb. 2015.

[4] A. Jain*, S.P. Ong*, G. Hautier, W. Chen, W.D. Richards, S. Dacek, S. Cholia, D. Gunter, D. Skinner, G. Ceder, K.A. Persson *(*=equal contributions)*

[5] IDAES, https://www.netl.doe.gov/research/coal/crosscutting/simulation-based-engineering